CMPUT 331, Winter 2025, Assignment 5: Frequency

Before beginning work on this assignment, carefully read the Assignment Submission Specifications posted on Canvas.

Introduction

Because mono-alphabetic substitution ciphers can be enciphered by more than 10^{20} possible keys, a brute-force approach will not work to decode a ciphertext. In this assignment, you will perform frequency analysis to decode mono-alphabetic substitution ciphers, and then evaluate the accuracy of a decoded plaintext.

All encipherment and decipherment in this assignment uses the simple substitution cipher. **Lowercase and uppercase letters are to be treated as the same. Punctuation, special characters, and spaces are not substituted by the mono-alphabetic substitution cipher.**

You will produce a total of five files for this assignment:

- **a5p1.py, a5p2.py, a5p3.py:** python solutions to problem 1, 2, and 3 respectively
- **plaintext.txt** containing your answer to problem 3
- a README file (**README.txt**)

Submit your files in a zip file named 331-as-5.zip. Lowercase and uppercase letters are to be treated as the same. You are not allowed to edit the import statements in any of the skeleton files. You are also not allowed to add additional helper functions to any of the files. Therefore all code should be written within the provided function skeletons.

Problem 1

Modify the provided skeleton file named *a5p1.py*, which contains functions *freqDict(ciphertext)* and *freqDecrypt(mapping, ciphertext)*.

When invoked with a text enciphered using the substitution cipher, the *freqDict(text)* function should perform frequency analysis on the entire text, using the frequency statistics methods from the textbook, and return a dictionary whose keys are the cipher characters, with the value for each key being the plaintext character it is assigned to using frequency analysis.

For English texts that have been deciphered using this *frequency analysis* method, the value of the most frequent cipher character should be 'E', and the value of the least frequent cipher character should be 'Z'. **If two or more letters occur the same number of times in the ciphertext, the letters that occur earlier in the alphabet should be considered to have a higher frequency.**

The *freqDecrypt(mapping, ciphertext)* function simply accepts a dictionary mapping of characters (in the format returned by "freqDict") and returns the resulting text after applying the mapping to each character in the ciphertext.

Note: This method of cracking the substitution cipher is far from perfect and it is unlikely to return the correct key.

We provide the letters in the alphabet sorted by the frequency in English (from Wikipedia's

article on Letter Frequency):

```
ETAOIN = "ETAOINSHRDLCUMWFGYPBVKJXQZ"
```

```
>> freqDict("AABBA")['B']
"T"

>> freqDict("good morning")['O']
"E"

>> freqDict("-: AB CD AH")['A']
"E"

>> freqDict("MKLAKAALK")
{'A': 'E', 'K': 'T', 'L': 'A', 'M': 'O'}

>> freqDecrypt({"A":"E","Z":"L","T":"H","F":"O","U":"W","I":"R","Q":"D"},
                "TAZZF UFIZQ!")
HELLO WORLD!
```

Problem 2

There are two primary ways of evaluating a solution to a simple substitution cipher: *key accuracy* and *decipherment accuracy*.

Key accuracy is the proportion of cipher character types in the alphabet which are mapped to their correct plaintext characters. Decipherment accuracy is the proportion of cipher character tokens in the ciphertext which are mapped to their correct plaintext characters.

Modify the provided skeleton file named *a5p2.py* that contains a function named *evalDecipherment(text1, text2)* where text1 is a plaintext and text2 is an attempted decipherment of a ciphertext created by enciphering text1. The evalDecipherment function should compare the two files, and return a list containing two fields that correspond to the key accuracy and decipherment accuracy of text2 w.r.t the plaintext, text1.

For example, if text1 contains the plaintext "this is an example", text2 might contain "tsih ih an ezample" – the text in text1 was enciphered, and text2 contains an imperfect attempt to decipher it. This decipherment has a key accuracy of 8/11, since there are 11 character types, and three of them, 'h', 's', and 'x', were deciphered incorrectly; it also has a decipherment accuracy of 11/15, since it is 15 characters long, but only 11 character tokens in the decipherment are correct. Thus, the function should return the list [0.7272727272727273, 0.7333333333333333]. **Your program should only count alphabetical characters in its decipherment evaluation and it should be case-insensitive.**

```
>> evalDecipherment("this is an example", "TSIH IH AN EZAMPLE")
[0.7272727272727273, 0.7333333333333333]

>> evalDecipherment("the most beautiful course is 331!",
        "tpq munt bqautiful cuurnq in 331!")
[0.7142857142857143, 0.625]
```

Problem 3

As we have seen, frequency analysis on real simple substitution ciphers is not as simple as assigning your most frequent letter to E, your second most frequent letter to T, and so on. Because real English texts rarely follow the exact theoretical frequency distribution, there is further complexity involved in deciphering simple substitution ciphers.

In this problem, you will write a function that computes a useful metric to help you decipher a simple substitution cipher, and then use it to break a real ciphertext.

Modify the provided skeleton code in *a5p3.py* which contains the function *checkLetterFrequency(ciphertext)*. The function should return a dictionary mapping letters in the ciphertext onto that letter's frequency in the text. That is, each letter is mapped to the number of times it appears in the ciphertext, divided by the number of alphabetic characters (letters A-Z, ignoring spaces and punctuation) in the text.

```
>> checkLetterFrequency("ABCD") == {"A" : 0.25, "B" : 0.25, "C" : 0.25, "D" : 0.25}
```

You will use this as a starting point to **manually** decrypt the ciphertext found in *ciphertext.txt*. You should start by comparing the frequencies reported by your code with the English letter frequencies provided in *frequencies.png*.

From there, you can make additional observations such as the fact that "a" and "I" are the only one letter words in English. You may also find it useful to analyze the frequency of bi-grams (pairs of letters) in English, and compare them with bigram frequencies in the ciphertext.

You should put the deciphered plaintext in a textfile called *plaintext.txt* and include it in your final submission. You will be marked on both the key accuracy and the decipherment accuracy of your proposed decryption. Do not worry about casing.